System Design­­­­­­­­

of

Prototype for self-Driving Car

Version 1.0

Prepared by

Shah Zain FA13BCS178

Ehtisham-ul-haq FA13BCS191

Nisar Khan FA13BCS196

Supervised by

Dr. Mubeen Ghafoor.

Department of Computer Science,

CIIT, Islamabad.

13th December, 2016.

**Table of Contents**

Table of Contents

[1. Introduction 2](#_Toc469388778)

[Scope 2](#_Toc469388779)

[Modules 2](#_Toc469388780)

[Image Acquisition and Processing: 2](#_Toc469388781)

[Lane Detection: 2](#_Toc469388782)

[Obstacles Detection: 3](#_Toc469388783)

[Car Navigation: 3](#_Toc469388784)

[Car Localization: 3](#_Toc469388785)

[Updating Target: 3](#_Toc469388786)

[Design Methodology and software process model 3](#_Toc469388787)

[Process Model 3](#_Toc469388788)

[2. Data Representation 4](#_Toc469388789)

[3. Process Flow/Representation 6](#_Toc469388790)

[4. Design Models 7](#_Toc469388791)

[State Transition Diagram 7](#_Toc469388792)

[System Block Diagram 8](#_Toc469388793)

[Data Flow Diagram 9](#_Toc469388794)

[Level 1 DFD 9](#_Toc469388795)

[Level 2 DFD 10](#_Toc469388796)

[Level 3 DFD 11](#_Toc469388797)

[5. Algorithm & Implementation 12](#_Toc469388798)

[De-noising (Smoothing) 12](#_Toc469388799)

[Canny Edge Detection 13](#_Toc469388800)

[Object Detection 15](#_Toc469388801)

[1- Feature-based object detection: 15](#_Toc469388802)

[Objection Tracking 16](#_Toc469388803)

[Image Subtraction for object tracking 17](#_Toc469388804)

[Object width calculation 18](#_Toc469388805)

[Motor Control 18](#_Toc469388806)

[Hough Line detection: 19](#_Toc469388807)

[Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. We will see how it works for a line. A line can be represented as y = mx+c or in parametric form, as \rho = x \cos \theta + y \sin \theta where \rho is the perpendicular distance from origin to the line, and \theta is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise ( That direction varies on how you represent the coordinate system. This representation is used in OpenCV). Check below image: 19](#_Toc469388808)

# Introduction

## Scope

The main components used will be the robot on top of which is a **mounted camera**, **LIDAR** and an **NVidia board** all of these would be connected and combined processing of these components would be able to make the robot work. There would be a predefined starting and ending points as we will tell the robot to move from its current position to a specific location on the map, car/robot will start moving from its starting point and will reach its destination (ending point). During its journey, video will be captured by the camera and will be processed on board; image by image. Different image processing techniques would be applied and any obstacles, lanes and objects i.e. person or animals will be detected on the way of its journey. On the basis of these processing results the car will take decision whether it should stop, increase the speed or decrease the speed etc. We still can take control of the car, if something goes wrong and requires manual operating. Car can face problems while operating in atmospheric conditions like rainy weather, windy day, snow and others.

## Modules

### Image Acquisition and Processing:

A camera mounted with robot is used to capture real time video. After capturing the video, this information will not be send to any external machine for processing and taking decision, rather the video will be processed within robot (onboard processing) which will reduce the time needed in image acquisition and processing.

### Lane Detection:

As the scene is continuously changing due to the movement of the car, this module will detect road lanes. Car is continuously processing image and checking the lanes and boundary of the road at every move. It will also detect when there comes a turn because at that point direction of lanes is not going to be straight.

### Obstacles Detection:

Our car will move on a specified path and will detect obstacles coming on its way. In this module objects detection methods are performed to identify the obstacles in its path. Obstacles can be cars, objects, persons etc. Car will adjust its speed accordingly.

### **Car Navigation:**

There would be a predefined station from where it is going to take the start. Location of car will always be known to us. Final point is given as destination at start.

### Car Localization:

Our car will keep track of its current location with the assistance of GPS. Whenever we want its location, the remote host PC will be used to make a request for its coordinates. Also, when the pursuit is stopped, our robot shall return to its Initial Designated Location using a path, calculated using car’s current position and its initial position. This navigation to and from these locations based on GPS will be autonomous.

### Updating Target:

The car is going to change its position consistently so images will be captured by camera and processed continuously. Lane and obstacles are detected at every single point.

## Design Methodology and software process model

We will use Procedural approach; we will develop each module separately, test them and integrate them.

### Process Model

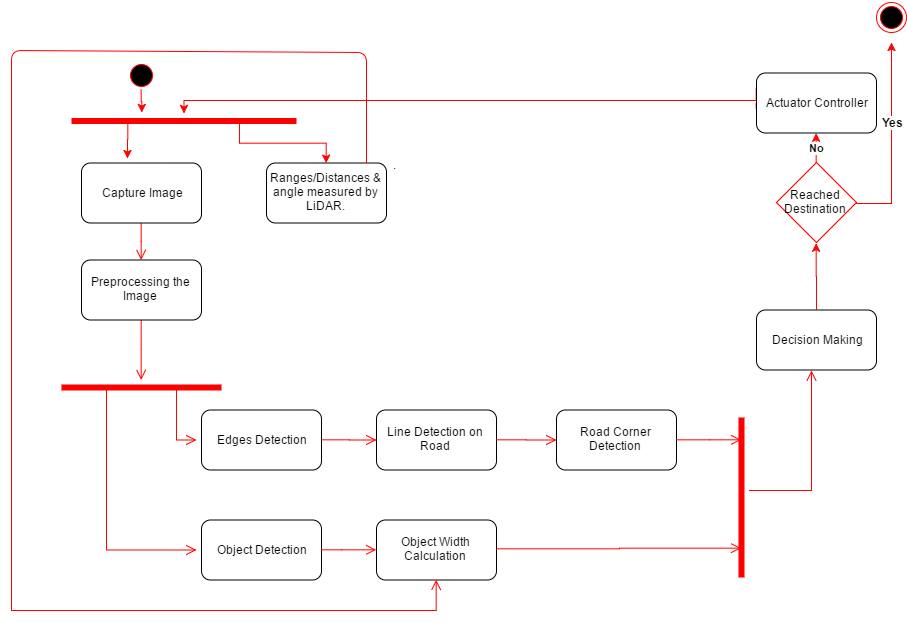
We will follow Incremental Process Model because we will develop our system incrementally, testing each module after its completion. For example, development of lane detection module and object detection module one by one.

# Data Representation

Most of our project is based on Image processing and robot control. In this case data, will be represented as follows:

|  |  |
| --- | --- |
|  | **NATURE OF DATA BEING PROCESSED** |
| Camera Image Acquisition | The type of data that is acquired through camera consist of 3-Dimensional array having values of Red Green and Blue in each dimension which is concatenated to get a color image. |
| LIDAR image acquisition | “LAS” is a published standard file format for the interchange of LiDAR data. Each LAS file contains metadata of the LiDAR survey in a header block followed by individual records for each laser pulse recorded. The header portion of each LAS file holds attribute information on the LiDAR survey itself: data extents, flight date, flight time, number of point records, number of points by return, any applied data offset, and any applied scale factor. The following LiDAR point attributes are maintained for each laser pulse of a LAS file: x,y,z location information, GPS time stamp, intensity, return number, number of returns, point classification values, scan angle, additional RGB values, scan direction, edge of flight line, user data, point source ID and waveform information.  **Intensity**: The return strength of the laser pulse that generated the LiDAR point.  **Return Number**: An emitted laser pulse can have up to five returns depending on the features it is reflected from and the capabilities of the laser scanner used to collect the data. The first return will be flagged as return number one, the second as return number two, and so on.  **Number of returns:** The number of returns is the total number of returns for a given pulse.  **Edge of flight line:** The points will be symbolized based on a value of 0 or 1. Points flagged at the edge of the flight line will be given a value of 1, and all other points will be given a value of 0.  **Scan angle:** The scan angle is a value in degrees between -90 and +90. At 0 degrees, the laser pulse is directly below the aircraft in our case it will be about 40-50.  **Scan direction:** The scan direction is the direction the laser scanning mirror was traveling at the time of the output laser pulse. A value of 1 is a positive scan direction, and a value of 0 is a negative scan direction. A positive value indicates the scanner is moving from the left side to the right side of the direction of robot, and a negative value is the opposite  All these combines to form a point cloud.  **Point Cloud:** Post-processed spatially organized LiDAR data is known as point cloud data. The initial point clouds are large collections of 3D elevation points, which include x, y, and z, along with additional attributes described above. |
| Preprocessing | During pre-processing stage, the data acquired through camera is passed through a Gaussian filter for denoising so that small edges that don’t contribute to the image are blurred and only prominent edges are retained. The image is also cropped so that only region of interest is selected and used. Data after this will be an array having intensity values at each point in x and y direction. |
| Line Detection | For line detection the input will consist of 2-D array that was output of preprocessing will be passed through prewet or Sobel filters by which only edges are restored which will be given as input to Hough transform that returns an array of rho and theta’s and the x,y coordinate where rho and theta are maximum will be the most prominent line in that image. We will then make line according to the formulae R=Xcos(theta)+Ysin(theta). |
| Object Detection | In Object detection, the type of data Given as input will be 2-D image data that will be processed and detected objects, i.e. output, consist of Dynamically changing list of sequence of objects which are the set of points, (x,y) ,where the object is detected. It consists of a structure having different data members used for storing data about detected object and its location. |
| Instructions to Robot | Based on the data that is collected in the above modules we will develop such techniques that will send appropriate Instructions to robot will consist of combination of LEFT, RIGHT, FARWARD and REVERSE. |

# Process Flow/Representation



**Explanation:** After turning on the robot, the camera and LIDAR will start acquiring image and the image from camera will be sent for preprocessing, in which denoising and downscaling is done, the resultant image is sent for edge detection and object detection through which objects and road lines are detected. After this, the decision will be made for robot movement by using the data from LIDAR and from processed image, this will be sent to the actuators which will make robot navigate its way. When the destination is reached, the process will stop.

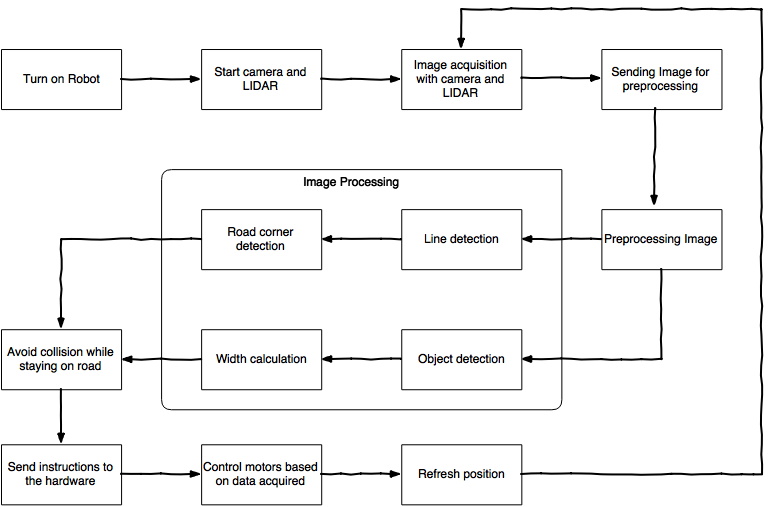
# Design Models

## State Transition Diagram



All transitions from one state to another will be based on the acquired image data.

## System Block Diagram



**Explanation:** After turning on the robot, the camera and LIDAR will start acquiring image and the image from camera will be sent for preprocessing, in which denoising and downscaling is done, the resultant image is sent for edge detection and object detection through which objects and road lines are detected. After this, the decision will be made for robot movement by using the data from LIDAR and from processed image, this will be sent to the actuators which will make robot navigate its way. When the destination is reached, the process will stop.

## Data Flow Diagram

### Level 1 DFD



### Level 2 DFD



### Level 3 DFD



# Algorithm & Implementation

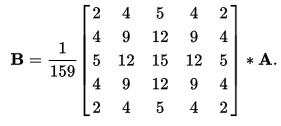
**Main algorithms used are given below:**

## De-noising (Smoothing)

These algorithms are applied in order to reduce noise and/or to prepare images for further processing.  
One of the efficient algorithm for de-noising an image is Gaussian Blur. Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function.

Where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

Here is an example of a 5×5 Gaussian filter, used to create the adjacent image, with σ = 1.4. (The asterisk denotes a convolution operation).



We can also use the 3×3 kernel. The larger the size of kernel, more smoothness of image.



## Canny Edge Detection

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed.

Canny edge detector includes:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Non-maximum Suppression
3. Hysteresis Thresholding

Since all edge detection results are easily affected by image noise, it is essential to filter out the noise to prevent false detection caused by noise. To smooth the image, a Gaussian filter is applied to convolve with the image. It is described earlier.

Derivatives (Change) are calculated in horizontal and vertical directions (using Sobel Operator). Then the gradient value is calculated. After calculating the gradient magnitude, thresholding is applied to keep most prominent edges, all other edges are discarded.

Compute gradient magnitude and orientation (Direction of edge).

### Non-Maximum Suppression

Non-Maximum suppression is applied to "thin" the edge. After applying gradient calculation, the edge extracted from the gradient value is still quite blurred. There should only be one accurate response to the edge. Thus non-maximum suppression selects the maximum valued point along normal to the edge direction.

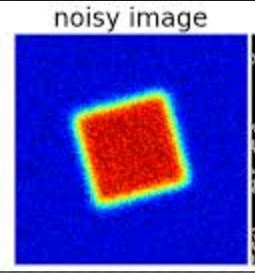
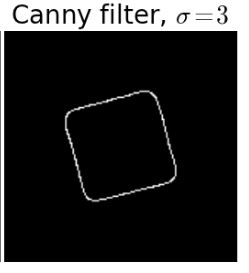
1. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction, the value will be preserved. Otherwise, the value will be suppressed.
3. Threshold the output non-maximum suppressed image

### Hysteresis Thresholding

The output of the non-maximum suppressed image are edges but since they are thresholded with single value there may be disconnected edges. To overcome this problem Hysterisis thresholding is used.

Steps:

* If the gradient at a pixel is above ‘High’, declare it an ‘edge pixel’
* If the gradient at a pixel is below ‘Low’, declare it a ‘non-edge-pixel’
* If the gradient at a pixel is between ‘Low’ and ‘High’ then declare it an ‘edge pixel’ if and only if it is connected to an ‘edge pixel’ directly or via pixels between ‘Low’ and ‘ High’

## Object Detection

Object detection is the process of finding instances of real-world objects such as faces, bicycles, and buildings in images or videos. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category. Objects are detected in several ways which are explained below.

### Feature-based object detection:

Objects are detected based on some features that are most prominent in the image. Like faces can be detected by detection of prominent face features like eyes and lips.



## Object Tracking

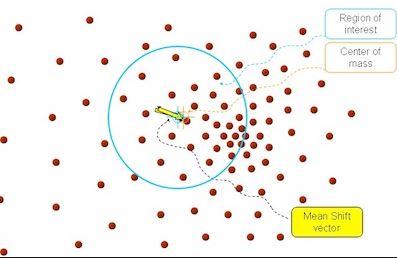
#### Mean Shift Algorithm

This section provides an intuitive idea of Mean shift and the later sections will expand the idea. Mean shift considers feature space as a empirical probability density function. If the input is a set of points then Mean shift considers them as sampled from the underlying probability density function. If dense regions (or clusters) are present in the feature space , then they correspond to the mode (or local maxima) of the probability density function. We can also identify clusters associated with the given mode using Mean Shift.

For each data point, Mean shift associates it with the nearby peak of the dataset’s probability density function. For each data point, Mean shift defines a window around it and computes the mean of the data point . Then it shifts the center of the window to the mean and repeats the algorithm till it converges. After each iteration, we can consider that the window shifts to a more denser region of the dataset.

At the high level, we can specify Mean Shift as follows :

1. Fix a window around each data point.   
2. Compute the mean of data within the window.   
3. Shift the window to the mean and repeat till convergence.



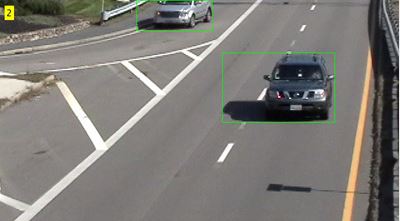
## Image Subtraction for object tracking

(it is used in both , object detection and object tracking)

* Image segmentation and blob analysis

Image segmented using background subtraction. The moving pixels (foreground) detected from the video frame above are shown in white.

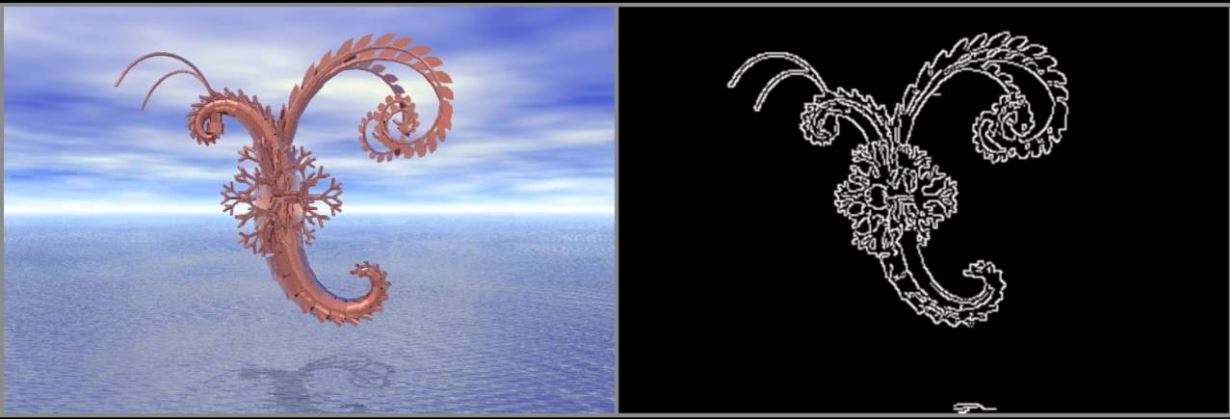
This is achieved by subtracting the first frame of video by its successor, in this way moving objects are detected and tracked.





## Object width calculation

Object width will be calculated by firstly by detection of object through image subtraction or any other object detection algorithm, after this we will know at what point in x, y coordinate the object is located. Then we will check the LIDAR image data at that point and see the distance of object and through this technique we’ll also calculate the width of object, this can be done concurrently with detection of edges to get better result overall. The result of the method explained through edge detection is give below:



## Motor Control

This algorithm will be used in controlling the motors from Arduino. For moving robot forward all motors are set to rotate at same RPM, for backward movement all motors need to move at same RPM but in reverse direction. For left, the front left and rear left motors need to turn slower than front right and rear right motor and for right movement, the front right and rear right motors need to turn faster than front right and rear right motor. By combination of these movements the robot will be controlled.

## Hough Line detection:

## Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. We will see how it works for a line. A line can be represented as y = mx+c or in parametric form, as \rho = x \cos \theta + y \sin \theta where \rho is the perpendicular distance from origin to the line, and \theta is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise ( That direction varies on how you represent the coordinate system. This representation is used in OpenCV). Check below image:

So if line is passing below the origin, it will have a positive rho and angle less than 180. If it is going above the origin, instead of taking angle greater than 180, angle is taken less than 180, and rho is taken negative. Any vertical line will have 0 degree and horizontal lines will have 90 degree.

Now let’s see how Hough Transform works for lines. Any line can be represented in these two terms, (\rho, \theta). So first it creates a 2D array or accumulator (to hold values of two parameters) and it is set to 0 initially. Let rows denote the \rho and columns denote the \theta. Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns. For \rho, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

Consider a 100x100 image with a horizontal line at the middle. Take the first point of the line. You know its (x,y) values. Now in the line equation, put the values \theta = 0,1,2,....,180 and check the \rho you get. For every (\rho, \theta) pair, you increment value by one in our accumulator in its corresponding (\rho, \theta) cells. So now in accumulator, the cell (50,90) = 1 along with some other cells.

Now take the second point on the line. Do the same as above. Increment the the values in the cells corresponding to (\rho, \theta) you got. This time, the cell (50,90) = 2. What you actually do is voting the (\rho, \theta) values. You continue this process for every point on the line. At each point, the cell (50,90) will be incremented or voted up, while other cells may or may not be voted up. This way, at the end, the cell (50,90) will have maximum votes. So if you search the accumulator for maximum votes, you get the value (50,90) which says, there is a line in this image at distance 50 from origin and at angle 90 degrees.

